Laboration 2

gean piere ventura cruz

2022-02-28

Dessa funktioner är tagna från labb 1 och kommer användas för att lösa fråga 2c. Den första funktionen multiplecerar matrisen P med sig själv n antal gånger. Den andra funktionen jämför två matrisers element och checkar ifall dessa är lika med varandra.

```
mpow <- function(P, n) {
  resultat <- diag(nrow(P))
  potens <- n
  while (potens > 0) {
    resultat <- P %*% resultat
    potens <- potens - 1
  }
  return(resultat)
}
```

```
matrices_equal <- function(P, Q, d = 4) {
  P_new <- trunc(P * 10^d)
  Q_new <- trunc(Q * 10^d)
  if (all(P_new == Q_new)) {
    return(TRUE)
  } else {
    return(FALSE)
  }
}
```

Uppgift 1

a. Vi kommer använda oss av “en_spelomgång” för att skapa funktionen “kim_spelar” som simulerar kims spelande. Funtionen kommer att returnera 1 om kim når 5kr, annars 0 om kim har slut på kapital.

```
en_spelomgång = function(p,kapital){
  if (runif(1) < p){
    return(kapital+1)
  } else{
    return(kapital -1)
  }
}

kim_spelar = function(p,kapital,mål=5){
  while(kapital > 0 & kapital < mål){
    kapital = en_spelomgång(p,kapital) #överföring till "en_spelomgång" som loppar spelet
  }
  if (kapital == 0){ #spelet når 0 kr
    return (0)  #går hem tomhänt
  } else{
    return (1) #går hem med vinst, 5kr
  }
}
```

b.

```
set.seed(960618)
sim_kim = function(p,kapital,n=1000){
  vinst=0
  for (i in 1:1000){ #loopar 1000 gånger
    vinst = vinst + kim_spelar(p, kapital, mål=5) #adderar vinst + ettorna från kim_spelar
  }
  return (vinst) #returnerar antal vinster
}
sim_kim(0.5,1)
```

```
## [1] 209
```

Vi har simulerat 1000 spelomgångar, och utav 1000 spel så vann hon endast 209 gånger vilket motsvarar ungefär 20% vinst och 80% förslust.

c.

```
set.seed(960618)
df <- data.frame(start_kapital = c(1:4))
df <- cbind(df, rbind(sim_kim(0.5, 1),sim_kim(0.5, 2), sim_kim(0.5, 3), sim_kim(0.5, 4)))
names(df) <- paste0(c("Kapital", "Vinster"))
knitr::kable(df)
```

Kapital	Vinster
1	209
2	392
3	586
4	825

tabell 1 visar fördelningen av startkapital till vänster och antal vinster till höger.

Först lägger vi märke till att när Kim väl har 0 startkapital då är hens chans att vinna 0%, och när hens startkapital är 5 då är hennes chans att vinna 100%. Sedan lägger vi också märke till att för kapital 2kr är chansen att vinna ungefär 40%. För kapital 3kr 60% och för kapital 4kr 80%. Vi ser ett mönster att startkapitalet/5 ger sannolikheten att vinna på Rodmans Roulette.

Uppgift 2

a. Vi ser Kims spelande som en MK med tillstånd 0-5 där 0 och 5 ses som absorberande tillstånd.

Vi skriver övergångsmatrisen P på standardformen bestående av 4 element:

$$P = \begin{matrix} P_T & R \\ 0 & P_R \end{matrix}$$

Vi får i postion 00 P_T. Denna matris är övergång mellan de transienta tillstånd. I postion 01 har vi R. Vilket är övergång från transienta tillstånd till rekurrenta tillstånd (antingen mål vinsten eller förlust av all kapital). I position 10 har vi 0. Då övergång från rekurrenta till transienta tillstånd sker. Slutligen i postion 11 har vi P_R. Det är övergång mellan rekurrenta tillstånd.

Återgår vi sedan tillbaka till P skrivet i “standardform”, så beskriver SR i matrisen P då n går mot oändligheten sannolikheten att från en transient i:te tillstånd hamna i en rekurrent j:te tillstånd (absorberande tillstånd i vårt fall).

SR kommer bli en matris av storlik 4x2. Den första kolumnen kommer då att ange sannolikheten att kedjan absorberas i tillstånd 5 givet att man börjar i någon tillstånd i. Den betyder att den andra kolumnen kommer att absorberas i tillstånd 0. Det element i matrisen SR beskriver sannolikheten att absorption sker i tillstånd 5 när spelaren startar med en krona blir då i vårt fall SR11.

b. Här ska vi skriva en funktion som tar en input som vinstsannolikhet och ger ut en övergränsmatris för denna vinstsannolikhet.

```
kims_matris = function(p,mål=5){
  q=1-p
  matris = matrix(0,                          #matris generator bestående av 0:llor
                  byrow=TRUE,
                  nrow = mål + 1,
                  ncol = mål + 1)
  #vi skapar våra absorverande tillstånd 0 och 5
  matris[1,1] = 1
  matris[mål + 1,
         mål + 1] = 1
  #fyller resterande luckor med p respektive q
  for (i in 2:mål){
    matris[i, i - 1] = q
    matris[i, i + 1] = p
  }
  return(matris)
}
```

c. Vi ska skriva en funktion som tar som input en övergångsmatris P skriven på standardform och ger ut som output motsvarande SR matris. Vi kan väl börja med en funktion som genererar matrisen P i standardform. Sedan använder vi oss av denna för att göra det enklare för oss att jobba med funktionen hitta_sr().

```
#vi använder oss av storleken av matrisen P för att sedan kopiera dess kolumner och klistra a
ntalet i momentary_matris 1. Sedan använder vi oss denna information för att kopiera raderna
och klistra dessa i momentary_matris2 som slutligen blir vår matris P.
P_standard = function(P){
  #storlek av den specifika matrisen P
  momentary_matris1 = matrix(0,
                             byrow = TRUE,
                             nrow = 6,
                             ncol = 6)
  momentary_matris2 = matrix(0,
                              byrow = TRUE,
                              nrow = 6,
                              ncol = 6)
  #kolumnenarna appliceras från matris P till momentary_matris1
  for(i in 1:5){
    momentary_matris1[i,] = P[i + 1,]
  }
  momentary_matris1[6,] = P[1,]
  #raderna appliceras från matris momentary_matris1 till 2 istället
  for(i in 1:5){
    momentary_matris2[,i] = momentary_matris1[,i + 1]
  }
  momentary_matris2[,6] = momentary_matris1[,1]
  return(momentary_matris2)
}
```

Om vi utgår ifrån att avgöra vilket n som Pn är ungeför lika med Pn+1 kan vi då sedan sortera ut matrisen SR från Pn. Vi kan då skria följande kodstycke.

```
hitta_sr = function(P_matris){
  #vi sätter upp våra variabler n, matris pn pch matris pn+1
  n = 0
  matris_pn = mpow(P_matris, n)
  matris_pn1 = mpow(P_matris, n+1)
  while (matrices_equal(matris_pn,matris_pn1,d=4) == FALSE){
    n = n+1
    matris_pn = matris_pn1
    matris_pn1 = mpow(P_matris, n+1)
  }
  sr = mpow(P_matris, n)[1:4, 5:6]
  return(sr)
}
```

Vi ska nu redovisa fallet för vilket n medför att Pn är ungefär lika med Pn+1 då p = 0.5.

```
kapital = 1:4 #endast dessa då vi redan vet vad som hände med 0 respektive 5

sanno_vinst = c(hitta_sr(P_standard(kims_matris(0.5)))[1, 1],
                hitta_sr(P_standard(kims_matris(0.5)))[2, 1],
                hitta_sr(P_standard(kims_matris(0.5)))[3, 1],
                hitta_sr(P_standard(kims_matris(0.5)))[4, 1])
antal_vinster = round(sanno_vinst * 1000)
tabell_2= data.frame(kapital, sanno_vinst, antal_vinster)
names(tabell_2) = paste0(c("startkapital", "sannolikhet att vinna 5kr", "lyckade spel"))
knitr::kable(tabell_2, digits = 3)
```

startkapital	sannolikhet att vinna 5kr	lyckade spel
1	0.2	200
2	0.4	400
3	0.6	600
4	0.8	800

tabell 2 visar fördelningen fördelningen av sannolikheter att nå 5 kr vinsten och där p=0.5. Kims kapital är uppdelad i 1-4 kr. Vi ser att hennes sannolikhet varierar beroende på ju mer kapital hon har. Samt att resultatet i tabell 2 inte riktigt stämmer med uppgift 1c. Det ser nästan ut som det har skett en avrundning då vi beräknar sr. Dock anser vi att denna form av simulering i 1c vara så pass nära sannolikheten för vinsten därmed är ett noggrant resultat.

Uppgift 3

Vi ska nu undersöka hur chansen att nå femkronorsnivån ändras när sannolikheten att förlora ett spel är större än chansen att vinna. Då chansen att nå 5 kr är p = 30%, 40%, 50%, 60% och 70%.

```
set.seed(960618)
n=1000
#metod 1 (simulering)
metod1_kim = c(sim_kim(0.3, 1, n) / n,
               sim_kim(0.4, 1, n) / n,
               sim_kim(0.5, 1, n) / n,
               sim_kim(0.6, 1, n) / n,
               sim_kim(0.7, 1, n) / n)
#metod 2 (sr avläsning)
metod2_kim = c(hitta_sr(P_standard(kims_matris(0.3)))[1,1],
               hitta_sr(P_standard(kims_matris(0.4)))[1,1],
               hitta_sr(P_standard(kims_matris(0.5)))[1,1],
               hitta_sr(P_standard(kims_matris(0.6)))[1,1],
               hitta_sr(P_standard(kims_matris(0.7)))[1,1])
metoder = data.frame(c("30%","40%","50%","60%","70%"), metod1_kim, metod2_kim)
names(metoder) = c("chans att vinna", "simuleringar", "sr avläsning")
knitr::kable(metoder, digits = 3)
```

chans att vinna	simuleringar	sr avläsning
30%	0.018	0.020
40%	0.077	0.076

chans att vinna	simuleringar	sr avläsning
50%	0.204	0.200
60%	0.367	0.384
70%	0.580	0.580

tabell 3 visar sannolikheterna för att nå 5 kr med olika värde på p (vinstchansen). Vi ser att resultatet skiljer sig minimalt och detta är förstås för att den ena metoden använder sig av simuleringar, medan den andra använder sig av teorin. Skulle vi utföra fler spel alltså större värde på n så skulle slutsatsen vara den följande: Sannolikheterna kommer att konvergera mot samma värde om vi utgår ifrån båda metoder och detta på grund av stora talens lag.

Uppgift 4

a. Det Robin gör är att hen satsar alla pengar hen har och om hen i ett visst ögonblick äger 3 eller kronor satsar hen alltså 2 respektive 1 krona.För detta kan vi skapa en Ivergångsmatris med sannolikheten p för vinst och 1-p för förlust.

```
generator = function(p){
  q = 1-p
  robin_matris = matrix(c(c(1, 0, 0, 0, 0, 0),
                          c(q, 0, p, 0, 0, 0),
                          c(q, 0, 0, 0, p, 0),
                          c(0, q, 0, 0, 0, p),
                          c(0, 0, 0, q, 0, p),
                          c(0, 0, 0, 0, 0, 1)),
                          nrow = 6,
                          ncol = 6,
                          byrow = TRUE)
  return(robin_matris)
}
```

b. Vi använder oss nu av sr matrisen från tidigare uppgift för att jämföra värderna mellan Robins och Kims matriser. Vi använder oss även av vinstchansen p = 30%, 40%, 50%, 60%, och 70%. Sedan ska vi redovisa värderna i en tabell.

```
kim_sr_värde = c(hitta_sr(P_standard(kims_matris(0.3)))[1, 1],
           hitta_sr(P_standard(kims_matris(0.4)))[1, 1],
           hitta_sr(P_standard(kims_matris(0.5)))[1, 1],
           hitta_sr(P_standard(kims_matris(0.6)))[1, 1],
           hitta_sr(P_standard(kims_matris(0.7)))[1, 1])

robin_sr_värde = c(hitta_sr(P_standard(generator(0.3)))[1, 1],
             hitta_sr(P_standard(generator(0.4)))[1, 1],
             hitta_sr(P_standard(generator(0.5)))[1, 1],
             hitta_sr(P_standard(generator(0.6)))[1, 1],
             hitta_sr(P_standard(generator(0.7)))[1, 1])
tabell3 = data.frame(c(0.3, 0.4, 0.5, 0.6, 0.7),kim_sr_värde , robin_sr_värde)
names(tabell3) = c("vinstchans","kim sr-värde","robin sr-värde")
knitr::kable(tabell3, digits = 3)
```

vinstchans	kim sr-värde	robin sr-värde
0.3	0.020	0.048

vinstchans	kim sr-värde	robin sr-värde
0.4	0.076	0.109
0.5	0.200	0.200
0.6	0.384	0.321
0.7	0.580	0.466

tabell 4 visar tolkningen av sr matrisen och oddserna att vinna 5kr för respektive vinstchans.

Som vi ser i tabell 4 så kan vi använda oss av Robins metod då vi utgår ifrån vinstchansen 30% och 40%. Kims metod kan vi använda oss av då vinstchangen överstiger 50% alltså 60% och 70%. Alltså kan vi spela som Robin (järvare metod) gör då p<50% och när p>50% väljer vi istället Kims (försiktigare metod) metod. Då p = 50% spelar det ingen roll vilken metod vi väljer. Vår tabell talar om för oss att båda metoder ger lika stor sannolikhet att vinna.